

Date:

Ex No :4 SIMULATION AND IMPLEMENTATION OF MUX/DEMUX

AIM : To simulate and implement 2X1 Multiplexer and Demultiplexer in FPGA using Verilog code in different modelings.

SOFTWARE USED:

- Xilinx 8.1 ISE Sim
- Spartan 3E

THEORY:

Multiplexer:

A multiplexer is a data selector device that selects one input from several input lines, depending upon the enabled, select lines, and yields one single output.

A multiplexer of 2^n **inputs has n select lines**, are used to select which input line to send to the output.

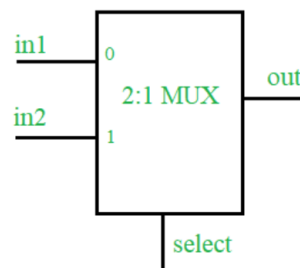


Fig: 2:1 Multiplexer

We'll code the 2:1 multiplexer in the following abstraction layers:

1. Structural modeling
2. Dataflow modeling
3. Behavioral modeling

1. Structural modeling:

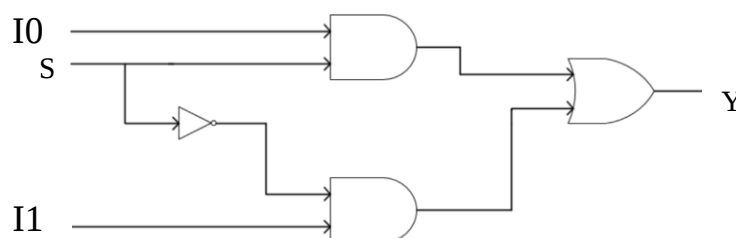


Fig: 2:1 MUX Verilog in structural modelling

In structural modeling, we describe the physical structure of a digital system. It is implemented using the logic gates in the circuit diagram. It gives us the internal hardware involved in the system. The gate-level modeling style uses the built-in basic logic gates predefined in Verilog.

2.Data flow modeling:

The dataflow modeling represents the flow of the data. It is described through the data flow through the combinational circuits rather than the logic gates used.

In Verilog, the assign statement is used in data-flow abstraction.

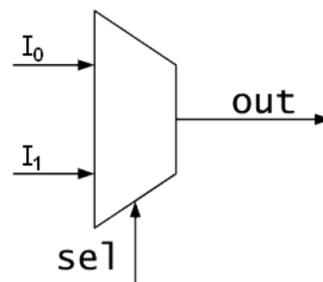


Fig. 2:1 MUX Verilog in Data Flow Model

3.Behavioral modeling:

The behavioral style, as the name suggests, describes the behavior of a circuit. It is the highest abstraction layer in the Verilog modeling of digital systems.

Demultiplexer:

In the 1 to 2 De-multiplexer, there are only two outputs, i.e., Y_0 , and Y_1 , 1 selection lines, i.e., S_0 , and single input, i.e., A . On the basis of the selection value, the input will be connected to one of the outputs.

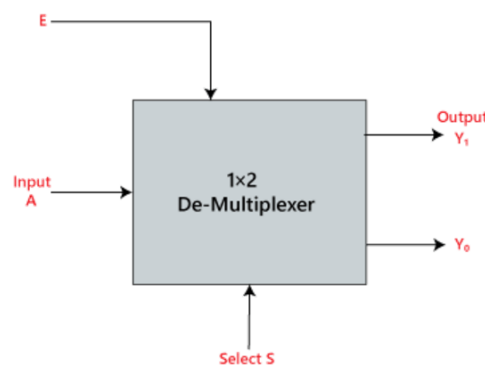
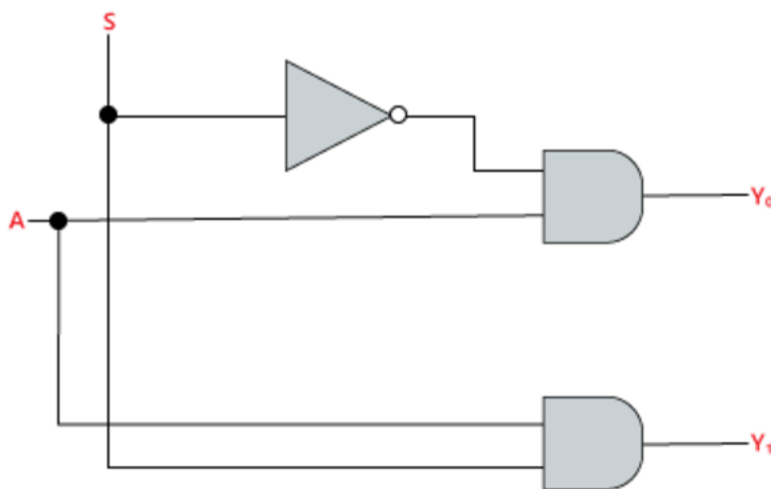


Fig. Block diagram

Truth Table

INPUTS	Output	
S_0	Y_1	Y_0
0	0	A
1	A	0

Logic Diagram



PROGRAM:

2:1 MUX in gate level model:

```

module mux_2x1_gl(input I0,I1,S,output Y);
wire sb,a,b;
not(sb,S);
and(a,sb,I0);
and(b,S,I1);
or(Y,a,b);
endmodule
  
```

2:1 MUX in Data Flow Model:

```

module mux_2x1_df(in0, in1,sel,out);
input in0, in1, sel;
output out;
assign out=(sel)? in1:in0;
endmodule
  
```

2:1 MUX in Behavioral Model:

```
module mux_2x1_b(in0, in1, sel, out);
input in0, in1, sel;
output reg out;
always @(*)
begin
    if(sel)
        out= in1;
    else
        out=in0;
    end
endmodule
```

1:2 DEMUX in gate level model:

```
module DEMUX_1_to_2( input s, input d, output y0, output y1);
not(sn,s);
and(y0,sn,d);
and(y1,s,d);
endmodule
```

1:2 DEMUX in Data Flow Model:

```
module demux_1_2( input sel, input i, output y0, y1);
assign {y0,y1} = sel?{1'b0,i}: {i,1'b0};
endmodule
```

1:2 DEMUX in Behavioral Model:

```
module1_2_DEMUX(input i0, input s0, output out1, output out2);
always @ (i0 or s0)
case (s0)
    0: out1 = i0;
    1: out2 = i0;
    default: out1 = 1'bx; out2 = 1'bx;
endcase
endmodule
```

RESULT: